# THE RTDCOM PROTOCOL - A REAL-TIME INTERCOMPUTER DATA TRANSMISSION PROTOCOL FOR THE ARPA NETWORK

Edwin W. Meyer, Jr.

Teledyne Geotech

Prepared for:

Air Force Technical Applications Center

20 May 1975

Unclassified

SECURITY CLASSIFICAT ON OF THIS PAGE (When Data Entered)

| REPORT DOCUMENTATION PAGE | | READ INSTRUCTIONS BEFORE COMPLETING FORM |
|---|---|---|
| 1 REPORT NUMBER<br>SDAC-TR-75-7 | 2 GOVT ACCESSION NO. | 3 RECIPIENT'S CATALOG NUMBER |
| 4 TITLE (and Subtitle)<br>THE RTDCOM PROTOCOL - A REAL-TIME INTERCOMPUTER DATA TRANSMISSION PROTOCOL FOR THE ARPA NETWORK | | 5 TYPE OF REPORT & PERIOD COVERED<br>Technical<br>6 PERFORMING ORG. REPORT NUMBER |
| 7 AUTHOR(s)<br>Meyer, Edwin W., Jr. | | 8 CONTRACT OR GRANT NUMBER(s)<br>F08606-74-C-0006 |
| 9 PERFORMING ORGANIZATION NAME AND ADDRESS<br>Teledyne Geotech<br>314 Montgomery Street<br>Alexandria, Virginia 22314 | | 10 PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS |
| 11 CONTROLLING OFFICE NAME AND ADDRESS<br>Defense Advanced Research Projects Agency<br>Nuclear Monitoring Research Office<br>1400 Wilson Blvd.-Arlington, Virginia 22209 | | 12 REPORT DATE<br>20 May 1975<br>13 NUMBER OF PAGES<br>31 |
| 14 MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)<br>VELA Seismological Center<br>312 Montgomery Street<br>Alexandria, Virginia 22314 | | 15 SECURITY CLASS. (of this report)<br>Unclassified<br>15a DECLASSIFICATION DOWNGRADING SCHEDULE |

16 DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

17 DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18 SUPPLEMENTARY NOTES

19 KEY WORDS (Continue on reverse side if necessary and identify by block number)

20 ABSTRACT (Continue on reverse side if necessary and identify by block number)

This document presents an ARPA Network Protocol for real-time high volume intercomputer data transmission utilizing prearranged non-switched data communication channels. This protocol deletes unnecessary and cumbersome aspects of the standard (NCP) ARPANET protocol and adds features intended to strengthen data integrity and to avoid error and loss due to network instability and outage. A special feature enables automatic switching of the data channel to alternate IMP ports in case a site changes its configuration. This protocol is suitable

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

for use with either the IMP Regular Message (type 0) or the newly introduced Uncontrolled Packet (type 3).

# THE RTDCOM PROTOCOL - A REAL-TIME INTERCOMPUTER DATA TRANSMISSION PROTOCOL FOR THE ARPA NETWORK

SEISMIC DATA ANALYSIS CENTER REPORT NO.: SDAC-TR-75-7

| | |
|---|---|
| AFTAC Project No.: | VELA VT/4709 |
| Project Title: | Seismic Data Analysis Center |
| ARPA Order No.: | 1620 |
| ARPA Program Code No.: | 3F10 |
| Name of Contractor: | TELEDYNE GEOTECH |
| Contract No.: | F08606-74-C-0006 |
| Date of Contract: | 1 July 1974 |
| Amount of Contract: | $2,237,956 |
| Contract Expiration Date: | 30 June 1975 |
| Project Manager: | Royal A. Hartenberger<br>(703) 836-3882 |

P. O. Box 334, Alexandria, Virginia 22314



///

## ABSTRACT

This document presents an ARPA Network Protocol for real-time high volume intercomputer data transmission utilizing prearranged non-switched data communication channels. This protocol deletes unnecessary and cumbersome aspects of the standard (NCP) ARPANET protocol and adds features intended to strengthen data integrity and to avoid error and loss due to network instability and outage. A special feature enables automatic switching of the data channel to alternate IMP ports in case a site changes its configuration. This protocol is suitable for use with either the IMP Regular Message (type 0) or the newly introduced Uncontrolled Packet (type 3).

TABLE OF CONTENTS

## ARPANET SUMMARY

The ARPA Network (ARPANET) is a complex store-and-forward message switching network implemented by a set of minicomputers (IMPs) interconnected by telephone lines. For our purposes each separate ARPANET communications path between two sites can be treated independently. This path consists of a pair of unidirectional message pipelines, one in each direction, through which individual data message packets can be pumped from the sender to the receiver.

Each message has a maximum length of 8095 bits* and at the IMP level is divided into two parts: a 32-bit IMP leader that contains routing information, and the body of the message, which is of variable length. The body is considered to be unstructured data at the IMP level, but is further subdivided into control and data fields at the RTDCOM protocol level.

See Heart, McKenzie, and BBN Report #1822 for detailed discussions of the ARPANET and the standard Host-Host Protocol.

---

*The discussion throughout this document assumes the use of the IMP regular message (type 0) as the transmission medium for RTDCOM data and control messages. However, the IMP Uncontrolled Packet (type 3), whose use is currently administratively restricted, seems more suitable for the purposes of the RTDCOM protocol, since it deletes features not needed by RTDCOM and is more expeditiously transmitted. The only differences that apply to the following discussion are that the Uncontrolled Packet has a maximum length of 1039 bits, as opposed to an 8095 maximum bit length for the Regular Message, and the scheme implied negative acknowledgements can not be used with Uncontrolled Packets.

## SUMMARY OF THE RTDCOM PROTOCOL

Although there may be many paths in the ARPANET simultaneously in use by a set of RTDCOM sites and many of the data flows along these paths may be related, the RTDCOM Communications Protocol considers each of these paths separately and independently. Moreover, each directional channel of the pair between the sites is treated as independent. For the purposes of discussion, only the channel in one direction between sites is described. The other channel works the same way but in the opposite direction.

Each channel can be considered to be a one-way pipeline down which data and control messages are passed, and also as a very narrow reverse pipeline for status information about whether the destination actually got the messages.

The basic cycle of sending a message from one RTDCOM site ("sender") to another ("receiver") is as follows:

Some module at the sending site (let us call it the "producer") gives the site's network module a batch of data to be sent to the other site. The network module takes this data and formats it into an IMP message, including a unique message number which the sender itself has assigned. The sender dispatches a copy of this message down the IMP channel but retains a copy in its internal buffers.

When the receiving site gets this message from the IMP channel, it dispatches an "acknowledgement" control message down the backchannel to advise the sender of correct receipt. This acknowledgement contains the message number of the message being acknowledged so that when the sending side gets it, it can find and delete the original message in its buffers.

A site need not wait for the acknowledgement for the current message to come back before dispatching the next one. The ARPA Network currently allows up to 8 messages to be simultaneously in transit from one site to another.* As long as a site can properly buffer and retransmit the pending messages, it can write-ahead up to this IMP-imposed limit.

---

* This restriction is separate for each site pair in each direction. Many hundreds of messages can be in transit at any one time in the entire network. The only penalty for attempting to exceed this limit is that the site-to-IMP channel blocks after reading the header of the ninth message and accepts the remainder only after one of the other messages has been delivered.

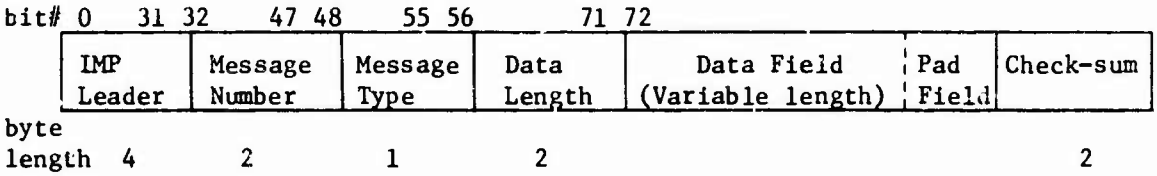
When the sender dispatches a message, it sets a timer; and if a certain length of time (the "timeout period" - several seconds) elapes without an acknowledgement being received for this message, the sender assumes the message was somehow lost in transmission. The message is retransmitted, and the timer is reset to timeout again if the acknowledgement still does not come in.

Because of this continual retransmission of unacknowledged messages, the receiver is assured of getting a second copy of messages lost in the network or those which the receiver itself discarded because they seemed garbled. Because of the message number, the receiver is able to discover and discard those messages which it has previously received correctly, so there is no confusion about transmitting multiple copies of a message.

## RTDCOM MESSAGE FORMAT

All messages generated by RTDCOM sites, whatever their purpose, are of basically the same format. Each consists of a 32-bit IMP leader followed by the message body, which is further subdivided into control and data fields by the RTDCOM protocol. The RTDCOM protocol defines an 8-bit byte length, and each field (except the Data and the Pad fields) breaks at byte boundaries. The Data field has an arbitrary bit length, so a Pad field follows it to pad it out to the next 8-bit boundary.

The RTDCOM message has the following format:




IMP Leader (4 bytes) - Its subfields are specified by the IMP-level protocol. It contains the host address of the RTDCOM site to receive this message and also a "message ID" field, which is renamed the Channel Number field for the purposes of the RTDCOM protocol. Each separate two-way network path between RTDCOM sites is assigned a different channel number, which is inserted into the Channel Number field of each message to eliminate confusion between separate channels.

Message Number (2 bytes) - This field has a different value for each separate data message, and uniquely identifies it among all others sent down the channel. It is used in the transmission error recovery system.

Message Type (1 byte) - This field is used to specify what type message (e.g. data or control) this is.

Data Length (2 bytes) - This field specifies the bit length of the Data field which follows immediately. A zero value is valid and is required by the definition of a few message types. It indicates that the Data and Pad fields are absent and that the Check-sum follows immediately.

Data Field (variable bit length) - Data or intersite commands. While this field is considered data of arbitrary values at the RTDCOM Communications

Protocol level, it is further structured at another modular level and may consist of intersite commands or data fields. This field and the Pad Field are absent if the Data Length Field equals 0.

Check-sum (2 bytes) - This is used for error detection at the receiving side. It is computed by the sender upon all other parts of the message except the IMP Leader each time the message is retransmitted.

Pad Field (variable bit length ranging from 1 to 7) - The Data field is of arbitrary bit length and may not end on an 8-bit byte boundary. The Pad field exists solely to pad out the Data field to the next byte boundary, and consists of all zero bits. This field is absent if the Data Length is evenly divisible by 8.

## CLASSES OF MESSAGES

The message type field ranges from 0 to 255. Of these, types 0 thru 127 are reserved as RTDCOM Communications Protocol control messages, and types 128 thru 255 are reserved for Data/Command Messages.

There are six different classes of messages, which the RDTCOM Protocol treats separately: The Data/Command messages and the five RTDCOM control messages ACK, START, RQSTART, HELLO, and IHY.

Data/Command Message (DCM) Class (types 128-255) - These message types are used to transmit data or intersite commands between RTDCOM sites. The message type codes and the Data Field may be specially interpreted at other modular levels of these sites, but within the RTDCOM Communications Module, all Data/Command Message types are treated identically and the Data Field is not inspected.

Acknowledgement (ACK) Message (type 0) - Used by the destination Host to acknowledge the correct reception of a DCM or Startup Message by the receiver (see Transmission Error Recovery section). The Data Length Field of this type message must be zero, and consequently the Data Field is absent. The Message Number Field of the message is set to that of the message being acknowledged.

Startup (START) Message (type 1) - A sender that wants to initialize its send channel issues this control message (see Channel Initialization section). The Data Field is absent.

Request Startup (RQSTART) Message (type 2) - A receiver that wants to initialize its receive channel issues this control message. It has the effect of inducing the sender to issue a START control message (see Channel Initialization section). The Message Number Field of this message type is unused and must be zero. The Data Field is absent.

HELLO Message (type 3) and I-Heard-You (IHY) Message (type 4) - At any arbitrary time a site may dispatch a HELLO message to the other side of the channel pair, which is to reply immediately with an IHY message. If somehow a site not concerned with the specified channel receives

this message it should ignore it. The Message Number field is used in a unique way as an aid to the Host Address Switchover Mechanism described later. The left-hand byte is set to the host address to which this message is sent (the right-hand byte is null), so that the receiver can determine its own host address. The Data field is absent.

## TRANSMISSION ERROR RECOVERY

There is a possibility that a message may be garbled or lost in network transmission between sites. Both problems are overcome by a retransmission upon timeout of acknowledgement scheme.

When a sending side transmits a Data/Command message, it must retain a copy of it until the receiving side transmits an Acknowledge message containing the number of the message. Upon receiving this ACK, the sending side can discard its copy. However, if the timeout period (several seconds long) elapses with no such acknowledgement, the sending site retransmits the message with the original message number and again begins timing, repeating the process until it receives an acknowledgement.*

When a site receives a Data/Command message, it has the option of ignoring it if it cannot handle it at that time. (The message will be retransmitted upon timeout.) Similarly, if the check-sum does not compute correctly, the message is ignored. If the check-sum does compute correctly, it is taken up and an Acknowledgement with the received message's Message Number is sent back.

If an Acknowledge message is somehow lost in transmission, at some later time the receiver will get a second copy of a message correctly received previously. Because of the Message Number, the receiver knows that this message is a duplicate. It sends back an ACK Message containing this Message Number to get the sender to purge its buffer of the message and stop resending it. The duplicated message itself is discarded.

---

* Use of the message sequencing option presented in the next section requires that the receiver get the complete sequence of transmitted messages, so the sender can not abandon a message except by reinitializing the sending channel.

## MESSSAGE SEQUENCING OPTION

While the transmission error recovery scheme assures that all transmitted messages are eventually received error-free, the mechanism for correcting transmission errors allows messages to be received out of order.

Gaps in the sequence appear if messages are lost in the network or rejected by the receiver, and retransmitted messages break up the sequence. At the receiving end the task is to reassemble the messages in proper sequence and discard duplicate messages that may get through due to lost ACK messages. This operation is the task of the communications level message sequencing option, intended as the normal but optional mode. Some paths may choose not to use it and perform this function at the data processing level. This possibility is discussed later.

The basic idea underlying the message sequencing scheme is that Message Numbers are assigned in ascending numerical sequence. At initialization time the sending site establishes an initial value and communicates this to the receiving site in a Startup message. Message Numbers of Data/Command Messages begin at the next value and ascend in sequence.

The conceptual mechanism of the message sequencing option is as follows:*

At the end site there is a Send Message Counter which contains the latest assigned Message Number. When a new (not retransmitted) Data/Command Message is to be transmitted, the Send Message Counter is incremented by one and the new value assigned as the Message Number of the new message.

---

* It is emphasized that all cells, buffers, and program structures presented in this protocol document are for purposes of illustration only. What this protocol prescribes is message flow behaviors at the site-site interfaces, and the internal structure is a matter of programming convenience. This is particularly true of the message sequencing option. The internal mechanism is so arranged as to segregate the communications function rigidly from the received data processing functions. As an actual programming matter, the message sequencing and buffering functions at the receiving end might be handled jointly by the communications and "processor" modules.

The receiving site procedure is more complicated: it involves a Receive Message Counter, which contains the Message Number of the last Data/Command Message of a completely received sequence of messages. It also contains an Out-of-Sequence Buffer of correctly received messages that can not yet be pieced into the complete sequence and given to the processor at this site.

A conceptual mechanism for handling the correctly-received messages using the Receive Message Counter (RMC) and Out-of-Sequence Buffer (OSB) is described below:

1) A Data/Command Message (DCM) correctly received from the network appears in a temporary buffer. Is its Message Number (MN) equal to or less than the RMC?
   YES: Send an ACK for this message, discard it, and EXIT.

2) Does its MN = RMC + 1?
   YES: Send an ACK for this message, pass it to the DCM processor, increment the RMC by 1, and proceed to step 5, below.

3) Does the OSB contain a message with this MN?
   YES: Send an ACK for this message, discard it, and EXIT.

4) Is the OSB full?
   YES: Discard this message and EXIT.
   NO: Send an ACK for this message, add it to the OSB, and EXIT.

5) Does the OSB contain a message with MN = RMC + 1?
   YES: Transfer the message with this MN to the DCM processor, increment the RMC by 1, and repeat this step.
   NO: EXIT.

The main disadvantage of the message sequencing option is that it is non-optimal for one special case: a path with firehose data flow where some data loss is acceptable and where there are limited buffers on the sending side. The objection can be illustrated in this manner: Suppose that the Send Buffers are completely full of unacknowledged messages when a new data packet comes in for transmission. In order to insure that the complete sequence of messages will be transmitted, the new data must be discarded

and the existing messages retained until acknowledgements arrive for them. This plan is suboptimal, because of the certainty that the new data packet will to be lost compared with the low probability of loss were it possible to delete an existing packet.

A more optimal scheme would be to delete the oldest existing message if the Send Buffers are full when a new data packet comes in, accepting the small probability that the message corresponding to the deleted packet has been lost in transmission. Such a thing can not be done unless the receiver does not depend on getting a complete sequence of message numbers, which is why message sequencing is optional.

The transmission regimen for the sending side is not altered in the absence of message sequencing in the normal case. A copy of a transmitted message is retained in the Send Buffer until its acknowledgement arrives, and the message is retransmitted upon timeout or arrival of an implied-NACK. If the Send Buffer is full when a new packet must be stored, the message with the lowest Message Number is selected for deletion, and the new packet is stored and transmitted. However, the old packet is immediately retransmitted before deletion (in order to improve chances for successful reception) if sufficient channel capacity exists.

The Send Message Counter and the sequential assignment of Message Numbers to outgoing messages is retained because a unique Message Number is required for associating an incoming ACK with a message copy in the Send Buffer and because the implied NACK facility requires an increasing sequence of ACKs.

If the message sequencing option is not used, the Receive Message Counter and Out-of-Sequence Buffer at the receiving end are deleted. Whenever a data/command message is correctly received, an ACK message containing the received Message Number is passed back, and the message is passed directly to the processor at the receiving end. It is presumed that the processor will be able to handle out-of-sequence, duplicate, and missing messages.

The use of the START and RQSTART channel initialization commands is deleted, because these are required primarily to synchronize the message counters at both ends. Moreover, there is no Receive Message Counter in the

absence of message sequencing. If the network goes down, the receiving side has to wait for data from the other end; while the sending side maintains its buffer full of the most recently generated messages, which are sent across when the communications link again opens.

## *IMPLIED NEGATIVE ACKNOWLEDGEMENT

An examination of the above scheme reveals that the sending side will get ACK messages with Message Numbers in increasing numerical sequence, ACKs for retransmitted messages excepted. A negative acknowledgement can be interpreted using this feature. Assume that two messages are sent, DCM(M) and DCM(M+1). If ACK(M+1) arrives without ACK(M) having previously arrived, ACK(M) will never arrive. ACK(M+1) can be treated as NACK(M) and used to trigger the immediate retransmission of DCM(M) without waiting for its timeout.

However, this simple scheme can be used only to trigger the first retransmission, since beyond the first implied-NACK most other ACKs have message numbers greater than that of the retransmitted message.

The following modification of this scheme will work for all retransmissions: Each copy of a transmitted message in the Send Buffer has a cell called the Retransmission Cell associated with it that contains the value of the Send Message Counter at the time of last transmission of this message. If an ACK with Message Number greater than this value comes in, it can be treated as a NACK for this message.

---

* This section does not apply if Uncontrolled Packets (type 3) are used as the IMP - network transmission mechanism.

## CHANNEL INITIALIZATION

Whenever a site wants to initialize its sending channel to another site, it sets its Send Message Counter (SMC) to some initial value and transmits a Startup (START) Message to the receiving side. The initial value of the SMC is used as the Message Number of the START. Upon correct receipt of the Startup Out-of-Sequence Buffer, the receiving side stores the Message Number in the Receive Message Counter and acknowledges the START. Not until the sending side receives an ACK message with the same Message Number as the START can it begin handling other traffic. The sender sets up a timer as with DM messages; and if it times out on the START, the Send Message Counter is incremented by 1. A START with the new Message Number is dispatched (to avoid confusion with the ACK of the failed START, should it later arrive), and the sending side begins timing on the acknowledgement for the new START. All other ACKs are ignored. (Other messages belong to the send path in the other direction.)

The START control message allows only the sending side to initialize a channel. To give the receiver the further ability to initialize the channel, the Startup Request (STARTRQ) control message is provided. The effect of the STARTRQ when sent by the receiver is to cause the sender to initialize the channel. Strictly speaking, the receiver can only request that the sender perform initialization.

Upon receiving a STARTRQ the sending side is obligated to initialize the channel by issuing a START message. To avoid regenerative loops, the sending side must ignore any STARTRQ it receives during the process of channel initialization (i.e. after it has dispatched a START and before the answering ACK has come back). The receiver sets up a timer for the STARTRQ control message and resends it if the timer runs out before a START control message arrives.

When should a channel be initialized? The basic rule is that a channel should be initialized whenever either side feels it is not co-ordinated with the other side. This is certainly true whenever a site begins operation or somehow destroys the contents of its internal cells, and also when it is

getting incomprehensible messages from the other side. Network failure, no matter of what duration, does not of itself require that the channel be reinitialized. If each side's cells remain valid, communication will resume normally when the network connection is restored.

## NETWORK FAILURE

At this protocol level, the failure of a channel connecting two sites is seen by the sending side merely as a failure to receive any acknowledgement for the messages it has transmitted. The receiving side stops getting messages, but does not necessarily view this condition as anomalous. This analysis indicates that network failure neither requires the channel to be initialized nor to be treated specially.

When the channel fails, the receiving side stops getting messages and the sending side stops getting acknowledgements. None of the cells and buffers on either side have been disrupted or discoordinated. If the send side merely adheres to its retransmission upon the timeout scheme, sooner or later the channel will be repaired and acknowledgements will begin to flow back. Transmission will thenceforth proceed normally.

No special programming is required to handle network failure. All that is necessary is for the sending and receiving communications modules to ignore all status messages that the IMPs return.*

If a network channel fails for an extended period, some sites might consider the contents of the send or out-of-sequence buffers as obsolete, preferring to delete them and reinitialize the channel. All that is necessary is to null the buffers and cells and begin the START or STARTRQ transmission. A retransmission-upon-timeout attempt will be made continually until the network channel reopens and the Startup handshake is successfully completed.

Depending on the implementation, a site might not interpret the IMP's responses (or lack of them) as indicating network failure. However, that criterion is not important. Are there items in the send or out-of-sequence buffers and has there been no traffic from the other side for some extended period? Then for all practical purposes, the network channel is down.

---

* At this protocol level only. IMP messages should be tallied or reported at some lower level within the communications module, but this problem is not a protocol issue.

## CHANNEL SWITCHOVER TO ALTERNATE HOST PORTS

Some sites might have alternate host ports or IMPs for use in case some change in machine configuration is necessary. The RTDCOM Protocol incorporates a host address switchover mechanism to adjust to a changed host port automatically with a minimum of channel outage and operator interference. The design goal is to allow a Host-IMP cable to be moved from one host port on an IMP to another at any arbitrary time and for the communications channel to switch over automatically to the new host address without data loss (assuming sufficient buffer capacity). This switchover capacity is also to serve in the case that one of the functional units (such as DP) is moved to backup hardware.

The basic philosophy behind the host address switchover mechanism is that it is the responsibility of a site to determine the new host address of the other side of the channel. It has stored within it a list of the possible host addresses of the other side, and whenever it gets an indication that the other side has changed host ports, it holds up normal traffic and sends out Hello control messages to all the possible addresses until it gets an I-Heard-You control message from one of them, which is then established as the new address of the other side.

It is desirable operationally to allow two different functional units to include the same host address as a possible alternate port. This ability can cause a problem, however, if one side of a channel mistakenly establishes communication with an entirely different functional unit which happens to be operating on one of the desired unit's possible host addresses. To eliminate this source of confusion, each separate two-way path over the ARPANET between the units of a co-operating system is assigned a different channel number. Each transmitted RTDCOM message carries its channel number in the "message identifier" subfield of the Host-IMP Leader. The channel number of every message that a site receives is inspected; and if for some reason it is for a different channel than that to which the receiver is attached, the message is discarded. After further hunting, the sender of that message will find and lock onto the correct host address.

Let us examine in detail the mechanism used to switch the channel if one side changes its host address. The switchover mechanism at each site uses these items: the Current Host Cell, the Possible Host List, and the Hunting Switch. The Current Host Cell contains the last known host address of the other side of the channel. The content of this cell is inserted into the Destination Host subfield of the Host-IMP Leader in each outgoing message during regular operation. The Possible Host List is an administratively specified list of the host addresses that the other side is allowed to use. The Hunting Switch is turned on whenever one side has switched addresses and the other is hunting for the new address.

Because a host need not know its own address to operate properly, the side which has changed host addresses need do nothing more than the ordinary routine. The unchanged side must do all the work. There are two ways by which a site changes its Current Host Cell to a new host address.

In the simplest way, the unchanged side receives a message from the new host address. Each time a message is received, its channel ID is checked. If it is not the expected value, the message is discarded. Next, the source host address is checked against the Current Host Cell. If they do not match, the other side has changed addresses, so the Current Host Cell is set to the new value.* This simple mechanism suffices for most cases of host address switchover.

However, this mechanism fails if the changed side is quiescent, in which case the unchanged side must actively hunt for the new host address. If a site sends a message to the old address of a site that has switched addresses, it will get one of two responses: one of the several kinds of "Destination Host Dead" status messages returned by the IMP, or no response at all (most likely because some other machine has attached to that address and ignores this message). Either case will cause the sending site to turn on its

---

* There is one exception that is very important in a few cases. If a Hello is received with the Source Host subfield of the IMP Leader equal to the first byte of the Message Number field (the host address of the receiver of the Hello), this site is talking to itself and its message must be ignored!

Hunting Switch and begin searching for the new address. The Hunting Switch is immediately turned on upon receipt of a "Destination Host Dead" IMP message and also upon the expiration of an extra-long timeout period (in comparison with the retransmission timeout) during which an expected response over the channel has failed to arrive.

When a site is in Hunting Mode, all normal traffic is held pending in its buffers; even channel initialization control messages are abeyant until the new host address is found. Instead a stream of Hello control messages, one for each site on the Possible Host List, is dispatched periodically until some message on this channel (probably an I-Heard-You reply) is heard from the other side. The receipt of any RTDCOM message with the right channel ID changes the Current Host Cell and turns off the Hunting Switch. All traffic held up during hunting can now be dispatched, and any messages lost in transmission to the old address will be retransmitted to the proper destination upon timeout of the acknowledgement.

There is no qualitative change in the host switchover mechanism described above if both sides of a channel simultaneously change host addresses, because it is not necessary for one side to have locked onto the other side's new address for that other side to achieve lock-on. All that is necessary is for each side to get a message through the new address of the other. Due to the Hello Fan-Out mechanism, each side will quickly lock onto the other side.

The Host Address Switchover mechanism, while integrated with the rest of the protocol at several points, ties in at a lower level and is reasonably independent of the other parts. It is the only level that must interpret IMP status messages. In normal operation it is invoked just before a message is dispatched and just after a message is received. It gains control and interrupts normal flow while in Hunting Mode. Other than for this special situation, it is totally disconnected from other aspects of the protocol, which need not be aware of its existence.

If one side of a channel goes down, the other side may well go into Hunting Mode. This activity is superfluous in two aspects: First, the down side will

probably come up again on its old address. This hunting is neither avoidable, given the need to elicit a response if the down side comes up at another address, nor serious, because real communication has been halted in any case. More seriously, a constant stream of Hello messages will add to network overhead while the other side is down. This problem can be mitigated by increasing the cycle time between the dispatch of Hello sets by a constant value in each cycle. With proper choice of initial cycle and increment, the Hello sets can initially be sent rather frequently in order to lock onto the other side immediately when it comes up. However, if the site remains down for an extended period, the cycle time will rapidly lengthen, reducing the network overhead to negligible amounts.

Both the Channel Reinitialization and Host Switchover mechanisms may be triggered if the other side of a channel goes down. This triggering causes no confusion, since these functions are independent and the Switchover mechanism has priority. Channel Reinitialization is not synchronous with Switchover, since it is properly triggered by a different event: the inability to maintain the Send Buffer.

## BUFFERING

The buffer capacity required at the sending and receiving sides of a RTDCOM channel depends on several factors: the data generation rate, the permissibility of sporadic data loss, the sender-to-receiver error rate, the round trip transit time (RTT) elapsed between dispatch of a message from a Sending Host and receipt of its ACK message, and the variability in the RTT. The first two factors are attributes of the data; the remainder, of the channel.

While we can precisely specify the data generation rate and allow a tolerable error rate, precise statistics concerning ARPANET transmission characteristics are not readily at hand, partly because the ARPANET is continually developing and partly because its characteristics are so variable. The overall result is that the few statistical experiments that we have made are not particularly relevant to our needs. However, a rough estimate of the behavior we can expect will be good enough for our current purposes. The statistics gained from actual operation will provide the best information for further refining the buffer and transmission strategy.

Two types of network parameters are important for transmission strategy planning: the end-to-end error rate and the RTT. No adequate estimate of the end-to-end error rate is available; however, it is highly dependent on the reliability of the connections at each end of the communications link. An error rate of one in ten thousand has been chosen, more or less arbitrarily. Some RTT measurements exist, and a mean value of a half second and a maximum reasonable value of five seconds have been chosen. Appendix A contains a brief synopsis of some error rate and RTT statistics measurements by ARPANET participants and gives a rationale for choosing the above values.

The buffer requirements of the sending and receiving sides vary greatly, depending on the worst case error rates and RTTs planned for. In general, the lengths of both the Send and Out-of-Sequence Buffers should be a number of elements at least equal to the number of messages transmitted during the maximum expected value of the RTT, which is about five. Of course the buffer requirements for paths that do not require completeness or use the message sequencing option are greatly relaxed.

A rough guide as to the number of cells required by the Send Buffer is the number of messages generated during the maximum expected RTT plus the expected maximum number of errors occuring within one RTT, or:

$$\text{BUFFERS} = (\text{RTT}_{max})/(\text{Message Rate}) + (\text{Errors}_{max}).$$

The rationale is that even without considering errors, the Send Buffer must be able to hold all of the messages generated between the time that the first one goes out and the time that the ACK for that message returns and frees space for another message. If a message must be retransmitted, it requires extra space from the time of the implied NACK until an ACK is received after retransmission--a period of $\text{RTT}_{max}$ seconds. The number of errors requiring retransmission during this period represents the number of additional buffers that must be planned for.

The receiving side is different. It must have enough cells in the Out-of-Sequence Buffer to contain the maximum expected incomplete sequence of message numbers, which is also related to $\text{RTT}_{max}$. If a message is not received and an implied-NACK sent back, the longest expected wait until the message arrives is $\text{RTT}_{max}$ (assuming that it is immediately retransmitted upon receipt of the NACK). A little analysis shows that the length of the OSB is relatively independent of the number of simultaneous gaps in the received sequence, but depends instead on the expected maximum number of retransmissions needed to transmit a message correctly:

$$\text{OSB} = (\#\text{retrans}_{max})* (\text{RTT}_{max})/(\text{Message Rate})$$

Note that the above calculations for both sending and receiving sides assume that retransmission of messages is triggered by implied-NACKs rather than timeouts.

What is to be done when the unexpected situation actually materializes and the buffers at the sending or receiving side fill up? The general procedure is to stop accepting messages from whoever is supplying them, letting the blockage ripple back towards the original producer, ultimately quenching its flow.

Consider the last step in the chain first: the receive end. Whether or not the message sequence option is used, there is some space at the receiving end used to hold incoming messages until they are processed. If this space is exhausted, the simplest and best thing to do is to stop accepting messages from the sender. The alternative is to delete, selectively, some messages already received. This option is unacceptable if completeness is required and redundant if message sequencing is not used.

The Send Buffer may fill due to transmission difficulties or the refusal of the receiving side to accept messages. The deletion algorithm for the No Message Sequencing Option has already been described. In other cases, completeness is presumably required, and the send side must stop accepting data from its producer. If the producer is not a "firehose" process, that is, if message flow can be quenched without losing data, there is no problem. The producer will halt and not resume until the sending side is again willing to accept data for transmission. The last case, requiring completeness of transmission from a firehose producer, given limited buffer space, is impossible to handle.

## MEASUREMENTS OF ARPANET TRANSMISSION CHARACTERISTICS

Two types of parameters are important in determining buffering strategy: the Roundtrip Transit Time (RTT - the time elapsed between dispatching a message and receiving an acknowledgement for it), and the rate at which messages are lost or garbled. There are published documents which give data on both items.

Kleinrock and Naylor (1974) report on IMP-IMP packet error rates during a week-long experiment. Of 86 interfaces in the network, more than half had error rates less than 1:100,000. Only 6 had error rates greater than 1:1000. The average error rate was 1:12,880. What we want is an estimate of the uncorrected end-to-end error rate, but the available figures represent IMP-IMP errors - which, moveover, are detected and corrected. We will derive a very uncertain value for the end-to-end error rate by assuming that the IMP-IMP error rate is of the same magnitude as the Host-IMP error rate (a rather tenuous assumption). Since the Host-IMP path has no error correction facility, it is assumed to be the major contributor to end-to-end errors, the rate of which we somewhat arbitrarily set at one in 10,000.

The Roundtrip Transit Time (RTT) is less easily arrived at. In particular, what we need is the maximum value we should reasonably plan for. H. Opderbeck reports (1974) an experiment in which one packet of messages was transmitted at maximum rate (RFNM-driven across a three-hop path). In that case the mean RTT was about a half second. Naylor reports (1973) on interarrival times (the same as RTT, but with different time boundaries) for various message lengths and IMP hops. For a maximum length message sent through six IMP hops, the following figures in milliseconds are given: 47.9 minimum, 1759 maximum, and 348 mean, with a standard deviation of 196. Naylor and Opderbeck (1974) give an extensive breakdown of mean RTTs catagorized by the number of IMP-IMP hops for actual network traffic during a period in December 1973. These RTTs ranged from 41 msec (0 hops) to 809 msec (13 hops). The mean was under 250 msec for less than 10 hops. Edward Taft (1974) reports

statistics indicating that during a one week pericd, the Harvard PDP-10 experienced an RTT of longer than 30 seconds once every 49,000 transmissions.

Given this data, it is reasonable to expect a mean RTT of about a half second, and a likely maximum value of a few seconds, say five.

The other factor determining buffering strategy is whether data loss is permissible. In some cases, small gaps in the data are tolerable, and buffer requirements are relaxed. On the other hand, where completeness of transmission is a design goal, the buffers must be large enough to handle the worst expected situation.

Buffering requirements are affected by the RTT and error rate in the following ways: First, the sending site must have enough buffers to store all messages generated during one RTT, i.e., the number of messages simultaneously in transmit and awaiting acknowledgements. Second, the sending site must also have enough buffers to handle the retransmitted messages until an acknowledgement arrives.

## REFERENCES


BBN Report #1822, 1974. Specifications for the interconnection of a host and an IMP. Cambridge, Mass.: Bolt Beranek and Newman Inc.

Heart, F. E., et al., 1970. The interface message processor for the ARPA computer network. Proceedings, 1970 Spring Joint Computer Conference. Washington: Tompson Book Co.

Kleinrock, L. and Naylor, W. E., 1974. On measured behavior of the ARPA network. Network Measurement Note #18. Menlo Park, Cal.: ARPA Network Information Center, Stanford Research Institute.

McKenzie, A., 1972. Host/Host protocol for the ARPA network. Menlo Park: ARPA Network Information Center, Stanford Research Institute.

Naylor, W. E., 1973. Real-time transmission in a packet switched network. Network Measurement Note #15. Menlo Park: ARPA Network Information Center, Stanford Research Institute.

Naylor, W. E. and Opderbeck, H., 1974. Network Measurement Note #19, RFC #619. Menlo Park: ARPA Network Information Center, Stanford Research Institute.

Opderbeck, H., 1974. Throughput degradation of single packet messages. RFC #632 Menlo Park: ARPA Network Information Center, Stanford Research Institute.

Taft, E., 1974. A few observations on NCP statistics. RFC #618. Menlo Park: ARPA Network Information Center, Stanford Research Institute.


RFC - Request for Comments.